Fermi General Purpose Grid Configuration

Joe Boyd, Ken Herner, Neha Sharma, Mike Kirby, Lisa Giacchetti, Miron Livny, Marco Mambelli, Margaret Votava, Anthony Tirandani

Draft Jan 12, 2016

Abstract

This document is meant to list the scheduling requirements for the general purpose grid cluster at Fermilab. It provides the batch processing resources for the intensity and cosmic frontier experiments at FNAL. Other collaborations use the cluster as well, primarily opportunistically.

Introduction

Fermilab has operated the GPGRID historically using a quota algorithm. Current monitoring has been developed with this viewpoint and our users and operations staff have been trained to look at current slot usage to understand if their newly submitted jobs should be running.

This configuration is becoming more unmanageable over time because

- We have been unable to successfully satisfy a user feature request that would allow an experiment itself (ie, no ops staff needed) to control the priority and precedence within its quota.
- 1 core/2GB slot configuration no longer readily satisfy the resources needs of the experiments. Experiments that use larsoft often need 4GB - 8GB of memory/core for certain production chains. The DES collaboration has specifically purchased large memory machines (128 GB) to handle image collating with processing requiring access to the full memory of the machine.

Requirements

Experiment Considerations/Challenges

- GPGrid contains about 16K cores. Experiments annually request their allocation with approximate sizing here. Currently, they make their requests using “standard” slots - 1 core/2GB each. The allocations range from 25 slots to approximately 2K slots. Within each allocation there are sub-allocations for groups. These allocations should be steady state and do not need to be maintained as immediately available. Job preemption is not foreseen as part of the GPGRID model.

- We support keep-up for many experiments. This often needs to run in a time window on the order of a day or two. Experiments have requested “reserved” slots on the order of 50-100 cores for short keep-up processing.
- Experiments want to dynamically control and change how jobs are ordered in their job queue - eg, pushing through a high priority analysis when needed.
- DES owns large memory nodes (128 GB) and they want them on demand. The time scale is a lead time of four (4) days.
- Many other experiments need some large memory slots, 4GB/core. In particular our top priority experiments - DUNE, NOvA, and MicroBooNE.
- Some experiments still can’t run on the OSG (MINOS+ and MINERvA)
- Ability to run multicore jobs is not yet needed, but may be on the horizon.
- Users want to monitor usage to verify they are getting what they requested. We need to be able to build monitoring tools that they will be able to interpret.
- The users should have tools available to profile their jobs for performance and matching based upon resources and possibly an email sent as the summary of every job that tells min, avg, max usage of memory, disk, and time will let the user do analysis of their own jobs.

Operational Considerations/Challenges

- Senior people can set the architecture, but the configuration needs to be maintained/triaged by more junior people.
- The architecture should be of a complexity such that most service providers can provide support. Seen as either a single moderate complexity system, or many smaller, identical configurations with changes in parameters.
- Current monitoring is centered on current slot count by experiment. We have little to no data with respect to memory usage. We can start collecting it, but it’s will take many months of collection to get any kind of analysis and then even longer to feed it back to experiments.
- Consideration must be given on how to value memory in job accounting. Monitoring the memory usage may allow to pair jobs with smaller memory footprint to the the ones with higher, using the nodes more efficiently. Extra jobs that fit on a node only because they use less memory should count the same as regular jobs using their slot or be "freebies" not affecting the quota?

- Operations staff also operates the CMS cluster and HEPCloud. The more similar the condor configurations are, the easier it will be to operate.
- The monitoring needs to provide metrics to determine usage efficiency of the system. This should be separable from the efficiency of the user jobs.

2.2 Things to think about from Miron

Here are some of the metrics that Miron is using at Wisconsin.

Wisconsin Monitoring

The gangliad/monitorinfd is pushing the history files on the schedd into a MongoDB so that you can process this information.
The startd has a history file with all the executions that ran on it. GLOW is collecting them every 24 hours (for all startd) and making a report.

3 design from user perspective

Things users care about

- Experiments want to understand if the resources made available will meet their needs. i.e.
  - following annual requests for computing resources how can experiments be confident that they are on track to recieve their allocation?
  - how can they be confident that the facility is capable of delivering resources to match their needs?
  - Will the accounting of utilized resources provide metrics that match requested resources with provided resources?
- system is designed to support their use case. The vast majority of resource requests that we see fall into one of these categories:
  - 1 core, 2 GB
  - 1 core, 4 GB
  - 1 core, 8 GB
  - DES — 1 core, 128 GB
- jobs start running in a reasonable timeframe
- what available resources request will get their jobs running right now
- jobs run successfully and finish in a reasonable timeframe
- In case of failures, they want
  - to know if resources they are submitting to are down
  - did the submission command make sense (did it request something impossible to ever match)
  - ability to do first level of debugging (such as condor_ssh_to_job)

    - tools/support personnel to provide clear explanation to help understand the cause of failure or inefficiency
- Straightforward control of relative priority of jobs within the experiment, even better if priority can be adjusted on jobs that have already been submitted.

4 design from operation perspective

GPGrid could be a single big cluster or multiple separate clusters where operators switch resources from one to the other (in HTCondor this is easy with condor_set_var). The former is more independent, the latter is less complex.

Keep in mind that partitioning resources in heterogeneous modules adds complexity and has a cost due mainly to fragmentation and draining:

- there are unused resources: a node may exhaust the cores or the memory or the pieces available on different nodes may be too small even if the aggregate is not
- there is cost in reassembling: to defragment a node it must be drained keeping cores and or memory unused
- longer jobs increase the time required for draining

These costs should be known should be clear if or how to account for them.

5 goals and metrics for success

A document clear enough to be understood and agreed upon by the experiments and the service providers should state the policies of GPGrid.
The metrics must verify that GPGrid is behaving as desired and if it is not help finding where is the problem:

- in the policy specification
- in the translation to the system configuration
- in the behavior of a component of the system

The metrics should also help using GPGrid efficiently.

Here some metrics useful to understand or steer GPGrid's behavior:

- Graphs per experiment that show their usage of the system
    - histograms of the CPU efficiency, memory, disk usage, and job time
    - is there a way to do time to match, and time to start?
- Graphs for the whole system showing time series graphs of the usage per experiment over time
- Time series showing the usage and differentiating the different type of slots (1core/2GB, 1core/4GB, …) used by each experiment

- Tables (from gratia data?) showing cpu-hours usage for each experiment over the last 7, 30, 90, and 365 days
- histograms showing the time left in each GPGrid glidein, requested time for each job, number of jobs that have matched/not matched, number of jobs that have matched but are idle due to failure to start due to time length not matching